Amendments to the Claims

Claims 1 - 24 (canceled)

Ε|

1   Claim 25 (currently amended): A method of searching and updating indexes to a data [[store]]

2   structure in a multi-processing environment, comprising steps of:

3   maintaining two indexes to [[the]] a single data [[store]] structure, a first index for

4   searching and a second index for updating;

5   responsive to each update of the second index, switching the indexes so that the first index

6   becomes the second index and the updated second index becomes the first index;

7   allowing searches that are in progress using the first index, before the switching, to

8   continue until completion after the switching, using ~~what is then~~ the newly-switched second

9   index;

10   after the switching, initiating new searches using ~~what is then~~ the newly-switched first

11   index;

12   when all searches in ~~what is then, after the switching, the~~ the newly-switched second index

13   have completed, updating ~~what is then~~ the newly-switched second index in an identical manner as

14   the update to which the switching step was responsive; and

15   preventing another operation of the switching step until completion of the step of updating

16   the second index in the identical manner.

Claim 26 (canceled)

Ɛ|

1   Claim 27 (currently amended): A program product storage medium containing computer

2   instructions that when executed in a computer perform a method of searching and updating

3   indexes to a data [[store]] structure in a multi-processing environment, the method comprising

4   steps of:

5   maintaining two indexes to [[the]] a single data [[store]] structure, a first index for

6   searching and a second index for updating;

7   responsive to each update of the second index, switching the indexes so that the first index

8   becomes the second index and the updated second index becomes the first index;

9   allowing searches that are in progress using the first index, before the switching, to

10   continue until completion after the switching, using ~~what is then~~ the newly-switched second

11   index;

12   after the switching, initiating new searches using ~~what is then~~ the newly-switched first

13   index;

14   when all searches in ~~what is then, after the switching, the~~ the newly-switched second index

15   have completed, updating ~~what is then~~ the newly-switched second index in an identical manner as

16   the update to which the switching step was responsive; and

17   preventing another operation of the switching step until completion of the step of updating

18   the second index in the identical manner.

Claim 28 (canceled)

1   Claim 29 (currently amended): A computer program product for serializing data structure

Serial No. 09/753,992

-3-

Docket RSW919990130US1

ℰ|

2   retrievals and updates in a multi-processing computer system, the computer program product

3   embodied on one or more computer-readable media and comprising:

4   computer-readable program code means for creating two identical data structures,

5   [[each]] both representing an initial state for accessing a single copy of stored data;

6   computer-readable program code means for performing searches against a first of the two

7   data structures, the computer-readable program code means for performing searches further

8   comprising a first program instruction for incrementing a search use count for the first data

9   structure atomically during each search to ensure no interference from other processes during that

10   search and a second instruction for decrementing the search use count for the first data structure

11   atomically after performing each search:

12   computer-readable program code means for performing a first update against a second of

13   the two data structures, yielding a revised data structure;

14   computer-readable program code means for switching the first data structure and the

15   revised data structure, responsive to completion of the computer-readable program code means

16   for performing the first update, such that the first data structure becomes the second data

17   structure and the revised data structure becomes the first data structure, the computer-readable

18   program code means for switching the data structures further comprising a third instruction for

19   re-ordering data structure pointers atomically to prevent interference from other processes during

20   operation of the computer-readable program code means for switching; and

21   computer-readable program code means for applying, after operation of the computer-

22   readable program code means for switching, the first update against the second data structure,

23   yielding a second data structure that is structurally identical to the first data structure;

𝑒𝑙

24   the computer-readable program code means for performing searches further comprising

25   computer-readable program code means for activating the computer-readable program code

26   means for applying the first update against the second data structure when the search use count

27   for the second data structure has a value indicating that no searches are being performed against

28   the second data structure.

1   Claim 30 (previously presented): The computer program product according to Claim 29, further

2   comprising:

3   computer-readable program code means for obtaining an exclusive lock on the second

4   data structure prior to operation of the computer-readable program code means for performing

5   the first update; and

6   computer-readable program code means for releasing the exclusive lock after operation of

7   the computer-readable program code means for applying the first update.

1   Claim 31 (previously presented): The computer program product according to Claim 29, wherein

2   the computer-readable program code means for performing the first update further comprises

3   computer-readable program code means for queuing a transaction that specifies one or more data

4   structure traversals and one or more data structure modifications that were performed to yield the

5   revised data structure, and wherein the computer-readable program code means for applying the

6   first update further comprises computer-readable program code means for performing the one or

7   more data structure traversals and the one or more modifications specified in the queued

8   transaction against the second data structure that results from operation of the computer-readable

Serial No. 09/753,992

-5-

Docket RSW919990130US1

$\mathcal{E}|\mathcal{C}|$

9   program code means for switching.

1   Claim 32 (previously presented): The computer program product according to Claim 29, further

2   comprising computer-readable program code means for performing a subsequent update against

3   the second data structure that results from operation of the computer-readable program code

4   means for applying the first update; and wherein operation of the computer-readable program

5   code means for performing the subsequent update causes another operation of the computer-

6   readable program code means for switching and the computer-readable program code means for

7   applying.

1   Claim 33 (currently amended): A computer system for serializing data structure retrievals and

2   updates in a multi-processing computer system, the computer system comprising:

3   means for creating two identical data structures, [[each]] both representing an initial state

4   for accessing a single copy of stored data;

5   means for performing searches against a first of the two data structures, the means for

6   performing searches further comprising means for incrementing a search use count for the first

7   data structure atomically during each search to ensure no interference from other processes

8   during that search and means for atomically decrementing the search use count for the first data

9   structure after performing each search;

10   means for performing a first update against a second of the two data structures, yielding a

11   revised data structure;

12   means for switching the first data structure and the revised data structure, responsive to

$\mathcal{E}$|

13 completion of the means for performing the first update, such that the first data structure becomes

14 the second data structure and the revised data structure becomes the first data structure, the

15 means for switching the data structures further comprising means for re-ordering data structure

16 pointers atomically to prevent interference from other processes during operation of the means for

17 switching; and

18   means for applying, after switching the data structures, the first update against the second

19 data structure, yielding a second data structure that is structurally identical to the first data

20 structure;

21   the means for performing searches further comprising means for activating the means for

22 applying the first update against the second data structure when the search use count for the

23 second data structure has a value indicating that no searches are being performed against the

24 second data structure.

1 Claim 34 (previously presented): The system according to Claim 33, further comprising:

2   means for obtaining an exclusive lock on the second data structure prior to operation of

3 the means for performing the first update; and

4   means for releasing the exclusive lock after operation of the means for applying the first

5 update.

1 Claim 35 (previously presented): The system according to Claim 33, wherein the means for

2 performing the first update further comprises means for queuing a transaction that specifies one

3 or more data structure traversals and one or more data structure modifications that were

ε|

4   performed to yield the revised data structure, and wherein the means for applying the first update

5   further comprises means for performing the one or more data structure traversals and the one or

6   more data structure modifications specified in the queued transaction against the second data

7   structure that results from operation of the means for switching.

1   Claim 36 (previously presented): The system according to Claim 33, further comprising means

2   for performing a subsequent update against the second data structure that results from operation

3   of the means for applying the first update; and wherein operation of the means for performing the

4   subsequent update causes another operation of the means for switching and the means for

5   applying.

1   Claim 37 (currently amended): A method for serializing data structure retrievals and updates in a

2   multi-processing computer system, comprising steps of:

3   creating two identical data structures, [[each]] both representing an initial state for

4   accessing a single copy of stored data;

5   performing searches against a first of the two data structures, the performing searches step

6   further comprising the step of incrementing a search use count for the first data structure

7   atomically during each search to ensure no interference from other processes during the search

8   and the step of decrementing the search use count for the first data structure atomically after

9   performing each search;

10   performing a first update against a second of the two data structures, yielding a revised

11   data structure;

Serial No. 09/753,992   -8-   Docket RSW919990130US1

Ɛ|

12   switching the first data structure and the revised data structure, responsive to completion

13   of the step of performing the first update, such that the first data structure becomes the second

14   data structure and the revised data structure becomes the first data structure, the step of switching

15   the data structures further comprising the step of re-ordering data structure pointers atomically to

16   prevent interference from other processes during operation of the switching step; and

17   applying, after the switching step, the first update against the second data structure,

18   yielding a second data structure that is structurally identical to the first data structure;

19   the step of performing searches further comprising the step of activating the step of

20   applying the first update against the second data structure when the search use count for the

21   second data structure has a value indicating that no searches are being performed against the

22   second data structure.

1   Claim 38 (previously presented): The method according to Claim 37, further comprising steps of:

2   obtaining an exclusive lock on the second data structure prior to performing the first

3   update; and

4   releasing the exclusive lock after applying the first update.

1   Claim 39 (previously presented): The method according to Claim 37, wherein the step of

2   performing the first update further comprises the step of queuing a transaction that specifies one

3   or more data structure traversals and one or more data structure modifications that were

4   performed to yield the revised data structure, and wherein the step of applying the first update

5   further comprises the step of performing the one or more data structure traversals and the one or

Serial No. 09/753,992   -9-   Docket RSW919990130US1

6   more data structure modifications specified in the queued transaction against the second data

7   structure that results from operation of the switching step.

1   Claim 40 (previously presented): The method according to Claim 37, further comprising the step

2   of performing a subsequent update against the second data structure that results from applying the

3   first update; and wherein the step of performing the subsequent update causes repeating the

4   switching step and the applying step.

1   Claim 41 (currently amended): A method for serializing data retrievals and updates in a

2   computing environment, comprising steps of:

3   creating two identical indexes, [[each]] both representing an initial state for accessing

4   stored data and each indexing a single copy of the stored data;

5   performing searches against a first of the two indexes;

6   performing a first update against a second of the two indexes, yielding a revised index;

7   serializing information describing on how the first update affected the second index,

8   including a traversal path taken through [[how]] the second index was traversed for making the

9   first update and one or more modifications made to [[how]] the second index was modified in the

10   first update;

11   switching the first index and the revised index, responsive to performing the first update,

12   such that the first index becomes the second index and the revised index becomes the first index;

13   applying, after the switching step, the first update serialized information to the second

14   index, using the serialized information about how describing the traversal path and the one or

Serial No. 09/753,992   -10-   Docket RSW919990130US1

ᘓ|

15   more modifications ~~the second index was traversed and modified~~ to ~~efficiently~~ traverse and

16   modify the newly-switched second index, thereby yielding a second index that is synchronized

17   with, and structurally identical to, the first index; and

18   · performing subsequent searches against the first index.

1   Claim 42 (currently amended): The method according to Claim 41, further comprising the step of

2   performing a subsequent update against the second index that results from applying the ~~serialized~~

3   ~~information~~ first update; and wherein the step of performing the subsequent update causes

4   repeating the serializing, switching, and applying steps.

1   Claim 43 (currently amended): A method of serializing access to data in a computing system,

2   comprising steps of:

3   maintaining two trees as indexes to [[the]] a single copy of data, a first of which is used

4   for searches and a second of which is used for update operations, each tree having a use count

5   associated therewith;

6   carrying out searches using the search tree, further comprising the steps of:

7   determining, for each new search request, which of the trees is currently the search

8   tree;

9   incrementing the use count for the search tree;

10   performing the new search request using the search tree; and

11   decrementing the use count for the search tree, responsive to completion of the

12   performing step; and

Serial No. 09/753,992   -11-   Docket RSW919990130US1

&I

13   carrying out each update using the update tree, further comprising the steps of:

14   determining which of the trees is currently the update tree;

15   performing an update ~~against~~ <u>to</u> the update tree;

16   serializing a record ~~of how~~ <u>describing</u> the update ~~affected~~ <u>to</u> the update tree;

17   switching the update tree to become the search tree and the search tree to become

18 the update tree, responsive to completion of the steps of performing the update and serializing the

19 record; and

20   applying the serialized record to the newly-switched update tree, provided that the

21 use count for the newly-switched update tree has reached a value that indicates that no search

22 requests are currently being performed against this newly-switched update tree, delaying the step

23 of applying the serialized record if necessary until the use count for the newly-switched update

24 tree has reached this value, and wherein the step of applying the serialized record ensures that

25 both the search tree and the update tree reflect each update.

1   Claim 44 (previously presented): The method according to Claim 41, wherein the indexes are

2   implemented as trees.

1   Claim 45 (previously presented): The method according to Claim 41, wherein the indexes are

2   implemented as hash tables.